

RDX2

X

```
runmodTemplate+ *j ;@ModName@ <- function(Times, newParms,
method="lsode",...){j  qj ldots <- list(...)^j  qj nstate <- @NSTATE@^j  qj nextra <-
@NEXTRA@^j  ↑ Outnames <- @OUTNAMES@^j  . ## Initialize parameters and state
variables^j  + Parms <- initparms(newParms,@ParmVector@)^j  † yinit <- Inity(Parms)^j
◀ y <- yinit[[1]]^j  † ydisc <- yinit[[2]]^j  % ## Put ydisc at the bottom of Parms^j
$ ## initialize the events structure^j  $ ev <- ConstructEvents(sort(Times))^j  1 ## insert the
events we already know about here^j  ⚡ @KnownEvents@^j  ◀ ## update 'now'^j  ◀ now <-
Times[1]^j  + ## setup the matrix to receive the output^j  - tmp <- matrix(nrow=0,
ncol=1+nstate+nextra)^j  ! ## run through the simulations.^j
repeat {^j  ← if (ev$quit(now)) break^j  % ys <- ev$doEvents(now,Parms,y,ev)^j  † y <-
ys[[1]]^j  qj ydisc <- ys[[2]]^j  Parms[names(ydisc)] <- ydisc^j  & nextint <-
ev$getNextInterval(now)^j  < tmp2 <- ode(y, nextint, "@DERIVS@",Parms, method=method,^j
F dllname="@DLLNAME@",initfunc="@INITNAME@",nout=nstate,^j  &
outnames=Outnames,...)^j  & y <- tmp2[nrow(tmp2),2:(nstate+1)]^j  now <-
tmp2[nrow(tmp2),1]^j  → tmp <- rbind(tmp,tmp2)^j  }^j  # tmp <- tmp[!duplicated(tmp[,1]),]^j
1 structure(list(result=tmp[tmp[,1] %in% Times,]^j  " parameters=Parms,^j
# model="@ModName@",^j  method=method,^j
control=ldots,^j  class="RDynOut")^j  }p
```